

- [67] S. Bhansali, A. Aris, A. Acar, H. Oz, and A. S. Uluagac, “A first look at code obfuscation for webassembly,” in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec ’22, (New York, NY, USA), p. 140–145, Association for Computing Machinery, 2022.
- [68] D. Genkin, L. Pachmanov, E. Tromer, and Y. Yarom, “Drive-by key-extraction cache attacks from portable code,” *IACR Cryptol. ePrint Arch.*, vol. 2018, p. 119, 2018.
- [69] G. Maisuradze and C. Rossow, “Ret2spec: Speculative execution using return stack buffers,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS ’18, (New York, NY, USA), p. 2109–2122, Association for Computing Machinery, 2018.
- [70] T. Rokicki, C. Maurice, M. Botvinnik, and Y. Oren, “Port contention goes portable: Port contention side channels in web browsers,” in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS ’22, (New York, NY, USA), p. 1182–1194, Association for Computing Machinery, 2022.
- [71] K. Pohl, G. Böckle, and F. Van Der Linden, *Software product line engineering: foundations, principles, and techniques*, vol. 1. Springer, 2005.
- [72] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. H. Tse, “Adaptive random testing: The art of test case diversity,” *J. Syst. Softw.*, vol. 83, pp. 60–66, 2010.
- [73] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, “Managing performance vs. accuracy trade-offs with loop perforation,” in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE ’11, (New York, NY, USA), p. 124–134, Association for Computing Machinery, 2011.
- [74] Avizienis and Kelly, “Fault tolerance by design diversity: Concepts and experiments,” *Computer*, vol. 17, no. 8, pp. 67–80, 1984.
- [75] G. R. Lundquist, V. Mohan, and K. W. Hamlen, “Searching for software diversity: Attaining artificial diversity through program synthesis,” in *Proceedings of the 2016 New Security Paradigms Workshop*, NSPW ’16, (New York, NY, USA), p. 80–91, Association for Computing Machinery, 2016.
- [76] B. Randell, “System structure for software fault tolerance,” *SIGPLAN Not.*, vol. 10, p. 437–449, apr 1975.
- [77] J. V. Cleemput, B. Coppens, and B. De Sutter, “Compiler mitigations for time attacks on modern x86 processors,” *ACM Trans. Archit. Code Optim.*, vol. 8, jan 2012.

- [78] A. Homescu, S. Neisius, P. Larsen, S. Brunthaler, and M. Franz, “Profile-guided automated software diversity,” in *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pp. 1–11, IEEE, 2013.
- [79] S. Bhatkar, D. C. DuVarney, and R. Sekar, “Address obfuscation: an efficient approach to combat a board range of memory error exploits,” in *Proceedings of the USENIX Security Symposium*, 2003.
- [80] S. Bhatkar, R. Sekar, and D. C. DuVarney, “Efficient techniques for comprehensive protection from memory error exploits,” in *Proceedings of the USENIX Security Symposium*, pp. 271–286, 2005.
- [81] K. Pettis and R. C. Hansen, “Profile guided code positioning,” in *Proceedings of the ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation, PLDI ’90*, (New York, NY, USA), p. 16–27, Association for Computing Machinery, 1990.
- [82] S. Crane, A. Homescu, S. Brunthaler, P. Larsen, and M. Franz, “Thwarting cache side-channel attacks through dynamic software diversity,” in *NDSS*, pp. 8–11, 2015.
- [83] M. T. Aga and T. Austin, “Smokestack: thwarting dop attacks with runtime stack layout randomization,” in *Proc. of CGO*, pp. 26–36, 2019.
- [84] S. Lee, H. Kang, J. Jang, and B. B. Kang, “Savior: Thwarting stack-based memory safety violations by randomizing stack layout,” *IEEE Transactions on Dependable and Secure Computing*, 2021.
- [85] Y. Younan, D. Pozza, F. Piessens, and W. Joosen, “Extended protection against stack smashing attacks without performance loss,” in *2006 22nd Annual Computer Security Applications Conference (ACSAC’06)*, pp. 429–438, 2006.
- [86] Y. Xu, Y. Solihin, and X. Shen, “Merr: Improving security of persistent memory objects via efficient memory exposure reduction and randomization,” in *Proc. of ASPLOS*, pp. 987–1000, 2020.
- [87] G. S. Kc, A. D. Keromytis, and V. Prevelakis, “Countering code-injection attacks with instruction-set randomization,” in *Proc. of CCS*, pp. 272–280, 2003.
- [88] D. Couroussé, T. Barry, B. Robisson, P. Jaillon, O. Potin, and J.-L. Lanet, “Runtime code polymorphism as a protection against side channel attacks,” in *IFIP International Conference on Information Security Theory and Practice*, pp. 136–152, Springer, 2016.